Adversarial Learning

Jie Tang

Tsinghua University

May 6, 2020

Overview

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Introduction

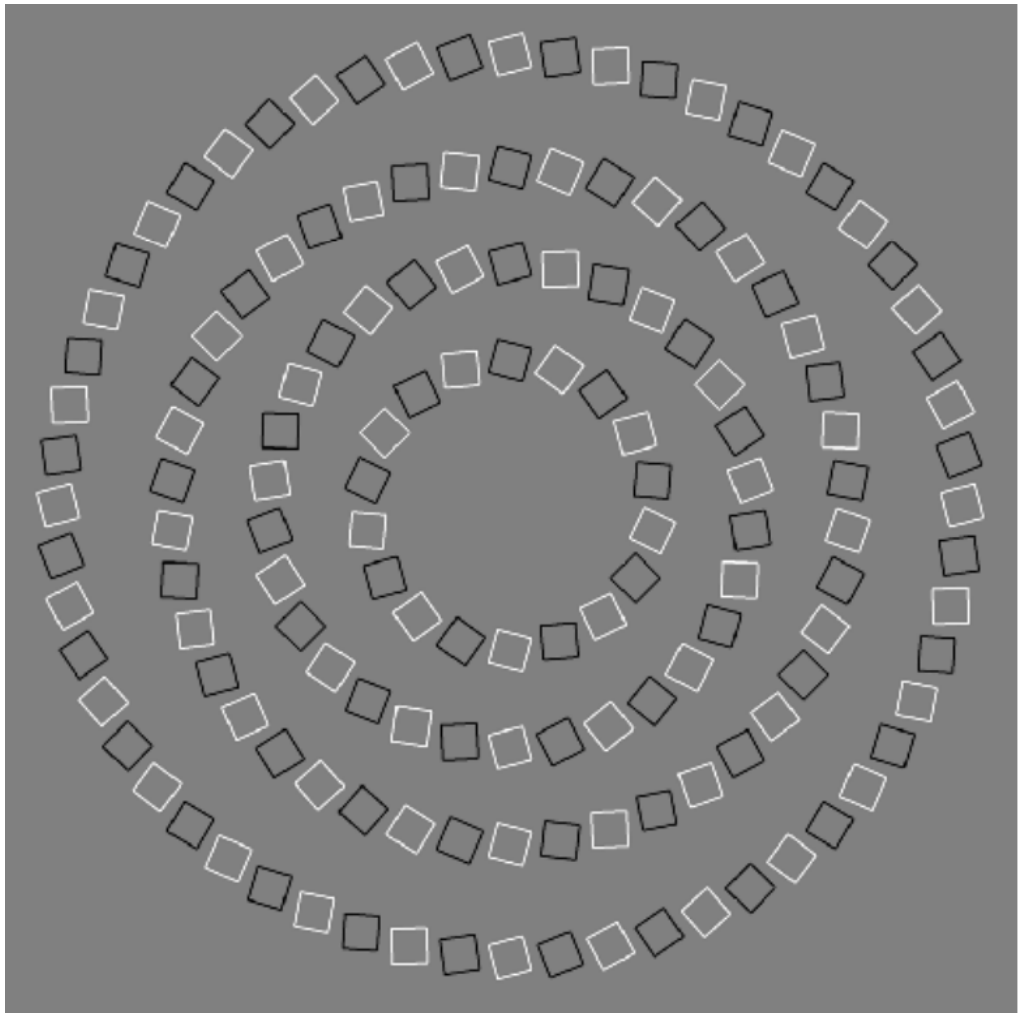
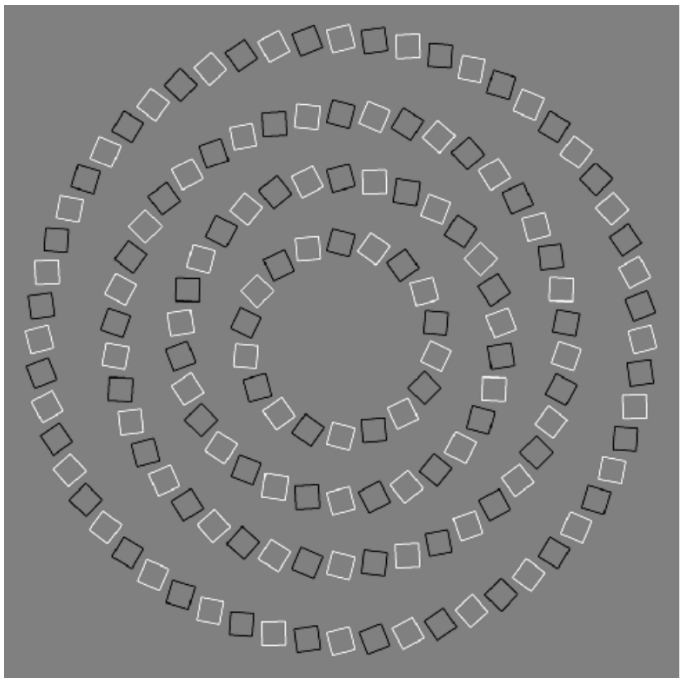


Figure 1: A figure that fools humans' eyes.

Introduction

- Most machine learning models assume a benign environment and pay little attention to outliers.
- There have been some works focusing on model training in malicious environment[1].
- This topic has become very hot in the deep learning era.



[1] Daniel Lowd and Christopher Meek. Adversarial learning. KDD'05.

Introduction

- A teaser: Given a network N pretrained on ImageNet, is there an image that would be classified as an iguana but actually a cat through human's eyes?
- The answer: Yes.
- The perturbed image is call an adversarial example, or simply adversary.
- This adversarial example is artificially constructed by attackers.
- How to construct and how to defend against adversaries is related to the research area of adversarial machine learning.



Figure 2: 92% cat



Figure 3: 94% iguana

Example



Original showing Alison Brie



Deepfake showing Jim Carrey instead of Brie



[2]

- DeepFake is able to replace a person's face in an existing image or video.

[2] Kietzmann, J.; Lee, L. W.; McCarthy, I. P.; Kietzmann, T. C. (2020). Deepfakes: Trick or treat?. Business Horizons. 63 (2): 135–146.

Introduction

- Traditional machine learning assumes all train and test examples are drawn independently from the same distribution.
- Recent years have seen many exciting models and their impressing results based on this assumption, many being deployed in real life system. To name a few applications, image classification, face detection, etc.
- Then why should we study adversarial machine learning?
- Security issues require moving beyond i.i.d. assumption, since the real life is anything but benign.
  - Not identical: attackers can use unusual inputs.
  - Not independent: attackers can repeatedly exploit one single mistake ("testset attack").
- Even under i.i.d. assumption, there exists adversarial examples due to overfitting or imperfect model.

Introduction

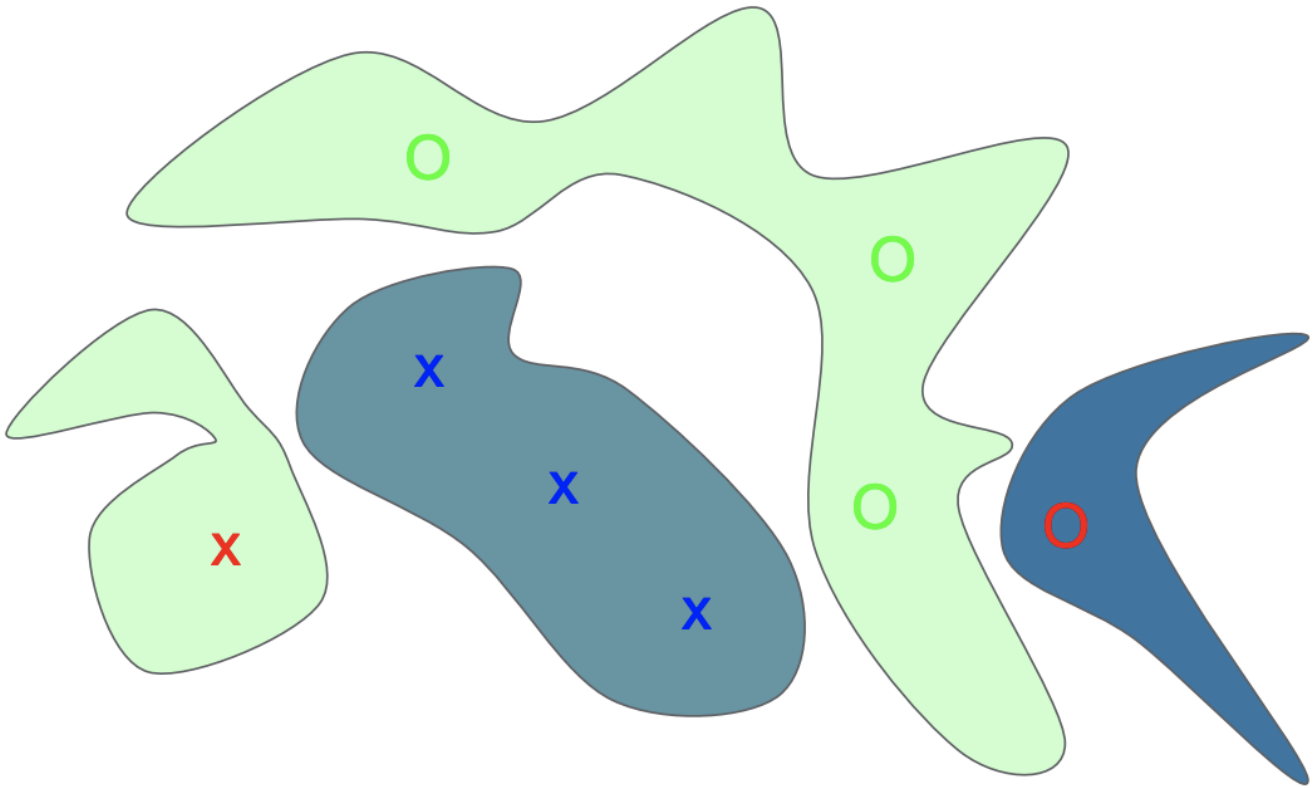


Figure 4: Adversarial example from overfitting.

Introduction

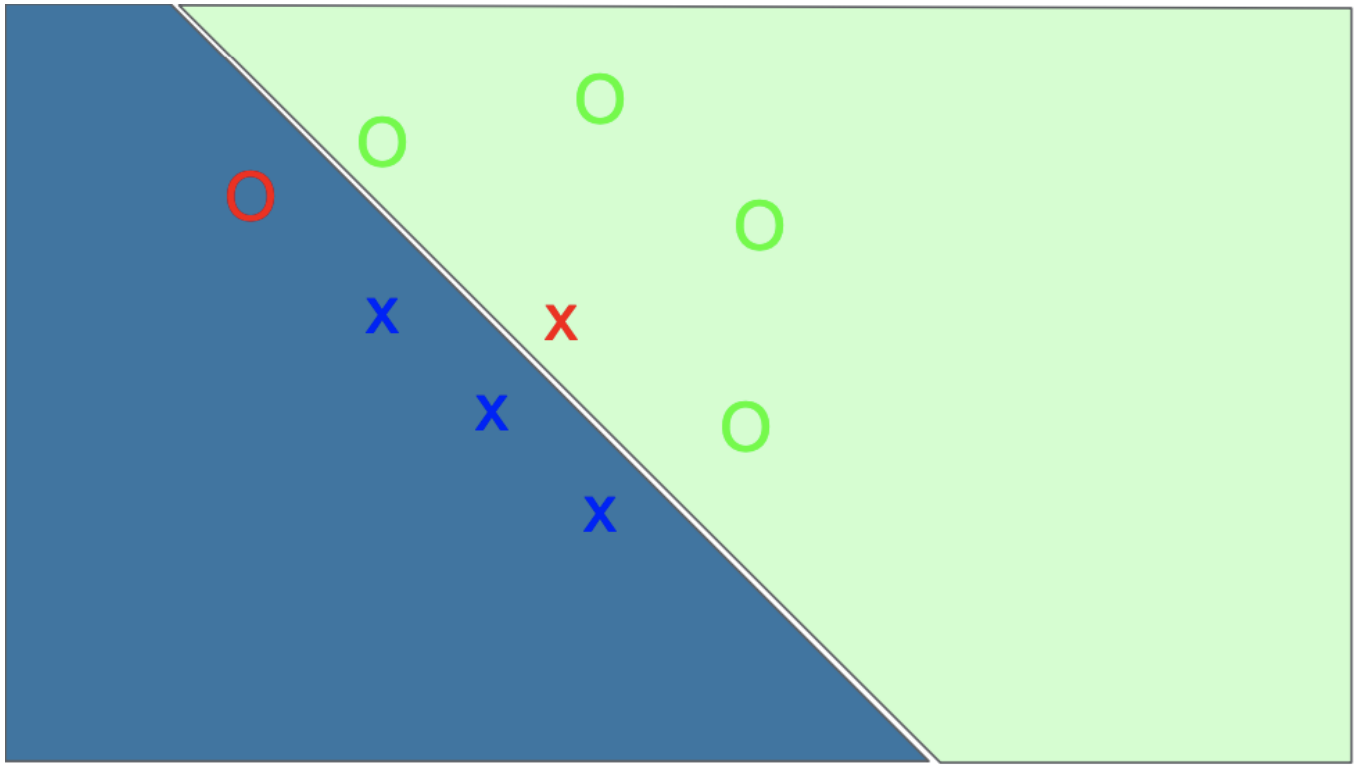


Figure 5: adversarial example from excessive linearity.

Introduction

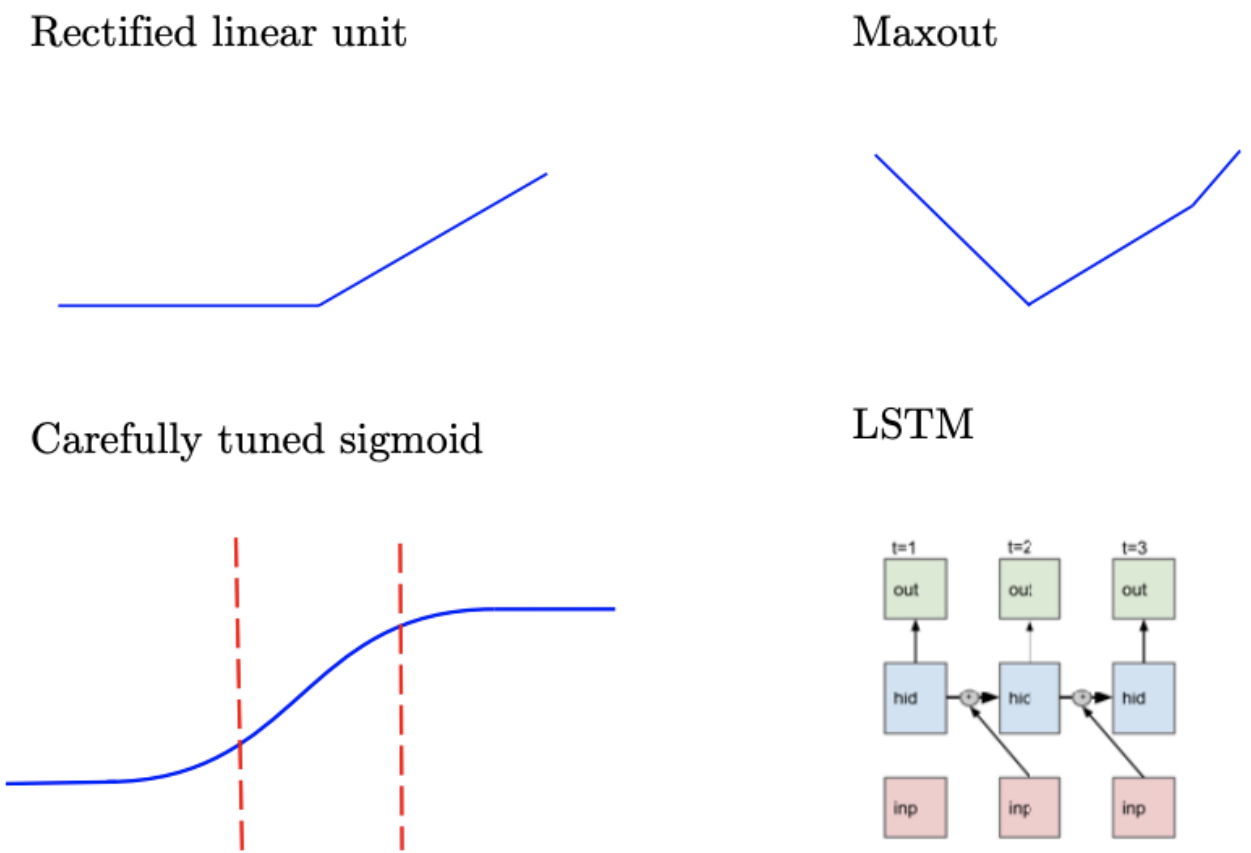


Figure 6: Modern neural networks are designed to behave linearly locally. This given an intuition explanation on why they can be attacked.

Attacking

- We have seen an example where a network trained on ImageNet is fooled.
- In fact, one can easily infer that not just neural networks can be attacked due to reasons we discussed above.
- We can attack Linear models (Logistic regression, softmax regression, SVMs), Decision trees and Nearest neighbors etc.
- After all, as the famous quotation by statistician George E. P. Box says, "All models are wrong, but some are useful."
- But how?
  - This depends on what the attackers know about the model.

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

White-box Attack

- White-box attack is a setting where attackers have full knowledge about the model targeted. In particular, they have access to model architecture and its parameters if it's a neural network being targeted.
- The goal of the attackers is to construct adversarial examples against the given model.

Fast Gradient Sign Method

- The fast gradient sign method (FGSM)[3] uses the gradient of the underlying model to find adversarial examples. It is fast because it can easily generate a lot adversarial examples due to its low computation cost.
- This method is described by

$$\mathbf{x}' = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)),$$

where J is the model parameterized by θ.

[3] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

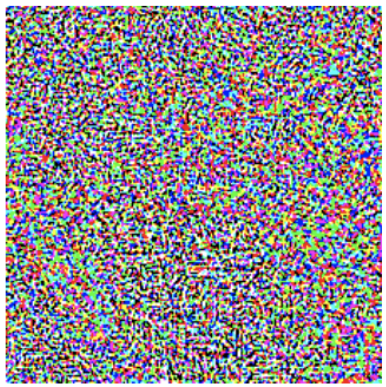
Fast Gradient Sign Method



$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$



$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Figure 7: An adversarial example of the GoogleNet. ϵ is set 0.007, which is the the magnitude of the smallest bit of an 8-bit image encoding after GoogLeNet's conversion to real numbers.

Fast Gradient Sign Method

- The efficacy of FGSM could be explained by a linear model.
- In many problems, the precision of an individual input feature is limited. For example, digital images often use only 8 bits per pixel so they discard all information below $1/255$ of the dynamic range.
- A very small perturbation at each pixel could lead to a large change of value of the output. For a linear model's adversarial example $\tilde{\mathbf{x}} = \mathbf{x} + \eta$, we have

$$\mathbf{w}^T \tilde{\mathbf{x}} = \mathbf{w}^T \mathbf{x} + \mathbf{w}^T \eta,$$

  all we need to do is to make $\mathbf{w}^T \eta$ as large as possible while keeping element of η small. This could be easily realized if the dimension of $\mathbf{x}$ is large enough.
- This explanation can be generalized to neural networks. Neural networks using LSTM, ReLUs, maxout are all intentionally designed to behave in very linear ways so as to enable easy optimization.

Basic Iterative Method (BIM)

- An iterative version of one-step attack FGSM [Kurakin 2016].
- It gives an iterative formulation to craft x':

$$x_0 = x$$
$$x^{t+1} = Clip_{x,\epsilon}(x^t + \alpha \mathbf{sign}(\nabla_x \mathcal{L}(\theta, x^t, y)))$$

- where *Clip* denotes the function to project its argument to the surface of $x's$ ϵ-neighbor ball $B_\epsilon(x) : \{x' : ||x' - x||_\infty \leqslant \epsilon\}$. (e.g. 1 unit of pixel change for each pixel)
- Heuristically searches the samples x' which have the largest loss value in the ball around the original sample x. Step numbers guarantee that the perturbation can reach the border (e.g. step $= \epsilon/\alpha + 10$).
- This iterative attacking method is also known as Projected Gradient Method (PGD) if the algorithm is added by a random initialization on x. [Madry et al., 2017]

Optimization-based Attack

- The attack given by FGSM misleads the given model, but not in a principled way. In other words, it says nothing about how the model would behave under its attacks, for example, which wrong class the model's output will be.
- What if we want not only fool the model, but also control its output?
- This could be realized by optimization[4].
- Still, we follow the principle that only the model is fooled, but we human can easily give the right response.

[4] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

Optimization-based Attack

- We formulate the problem as follows. Denote by $f : \mathbb{R}^m \to \{1, 2, ..k\}$ a classifier mapping image pixel value vectors to a discrete label set. We assume the associated loss functional $L_f : \mathbb{R}^m \times \{1, 2, ...k\} \to \mathbb{R}^+$ of f is continuous. For a given $x \in \mathbb{R}^m$ and the targeted label $l \in \{1, 2, ..k\}$, we aim to solve the following constrained optimization problem:

$$\min \|r\|_2^2$$
$$\text{s.t.} f(\mathbf{x} + r) = l.$$

  Then $\mathbf{x}' := \mathbf{x} + r$ is an adversarial example. (Note the optimization problem is nontrivial only when $f(x) \neq l$).

- This problem can be hard to solve. By Lagrange method, we write the constraint as penalty in the objective

$$\min c\|r\|_2^2 + L_f(\mathbf{x} + r, l). \tag{1}$$

  Then we can approximately solve this by gradient method.

An Example

- Formulation: Find x such that $\hat{y}(x) = y_{iguana} = (0, 1, 0, ...0)^T$ where $x = x_{cat}$.
- Loss is

$$L(\hat{y}, y) = \frac{1}{2}\|\hat{y}(W, x) - y_{iguana}\|_2^2 + \lambda\|x - x_{cat}\|_2^2.$$

- Optimization is leading by gradient of the image instead of the parameter, i.e.,

$$\frac{\partial y(W, x)}{\partial x}$$

  instead of

$$\frac{\partial y(W, x)}{\partial w}.$$

- One can choose different λ or different norm to get different result.

Deep Fool

- Find a path such that x can go beyond the decision boundary, so that the classifier will give a different prediction for x.
- E.g. to attack x_0 (true label is digit 4) to digit 3, the decision boundary described as $\mathcal{F}_3 = \{z : f(x) = F(x)_4 - F(x)_3 = 0\}$. In each attacking step, it linearizes the decision boundary hyperplane using Taylor expansion $\mathcal{F}'_3 = \{x : f(x) \approx f(x_0) + \langle \nabla_x f(x_0)(x - x_0) \rangle = 0\}$, and calculates the orthogonal vector w from x_0 to plane $\mathcal{F}'_3$.
- This vector w can be perturbation that makes the x_0 go beyond the decision boundary $\mathcal{F}_3$. By moving along the vector w, the algorithm is able to find the adversarial example x'_0 that is classified to class 3.[5]

[5] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. CVPR'16.

Deep Fool

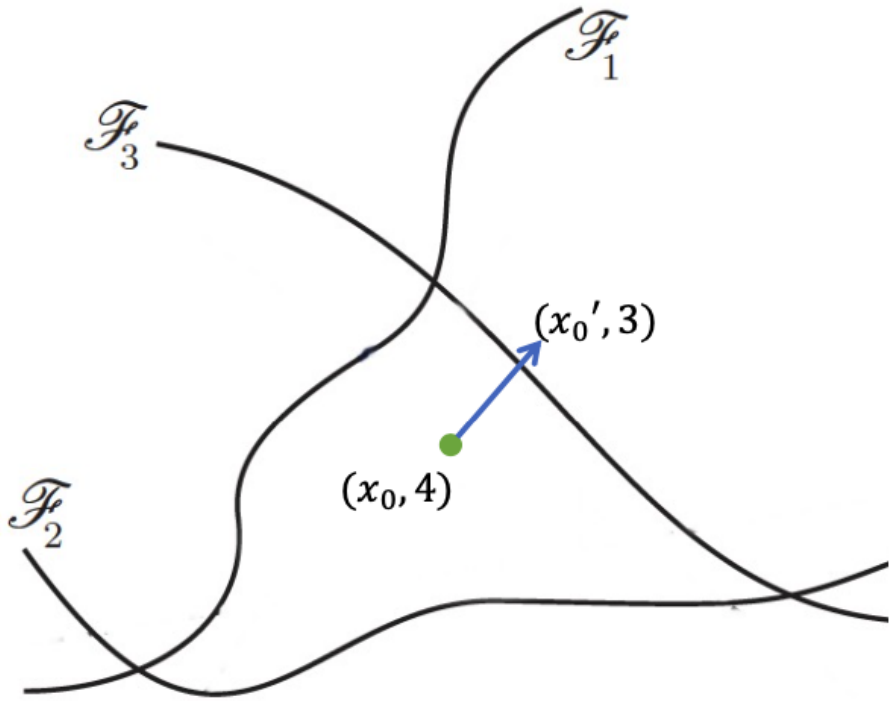


Figure 3. Decision Boundaries: the hyperplane $\mathcal{F}_1$ ($\mathcal{F}_2$ or $\mathcal{F}_3$) separates the data points belonging to class 4 and class 1 (class 2 or 3). The sample x_0 crosses the decision boundary $\mathcal{F}_3$, so the perturbed data x_0' is classified as class 3. (Image Credit: (Moosavi-Dezfooli et al., 2016))

For common DNN image classifiers, almost all test samples are very close to their decision boundary.

This suggests that the DNN classifiers are not robust to small perturbations.

Disccusion about White-box Method

- According to the linear explanation of FGSM, it is obvious that FGSM's efficacy is not unique to neural models. In fact, all parameterized model may suffer from sufficient feature dimensions and limited precision, thus be vulnerable to FGSM.
- Both FGSM and optimization-based attacks require the gradient of the model w.r.t. the input. This has two implications.
  - The models vulnerable to these attacks should be continuous and further, differentiable.
  - For all differentiable models, the harder to take derivatives or optimize, the harder to attack them with FGSM or optimization-based attacks. Any model with a linear or convex loss functional is vulnerable, while highly non-linear models such as neural networks are harder to attack.

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Black-box Attack

- Both FGSM and optimization-based attacks are white-box attacks, since these methods require knowledge about the model's internal, for example, the parameters of the model.
- In the setting of black-box attack, the internal information is not available to attackers. The only information available is the output of the model for any given input.
- Is intended attacks still possible in this case?
- First intuition: at least some random input will cause the model to behave badly. After all, there's no perfect model in the world.

Zero-query Attack

- In some cases, the attackers have only limited access to the input and output of the targeted model.
- In the most extreme case, the attackers have no access, and thus he can only guess an adversarial example.
- Random perturbation to normal input would be an easy starting attack strategy.
- Is there any prior to guide the guess?
- Transferability-based attack[6] gives an answer. We can use adversarial examples of a known model to attack an unknown model.

[6] Liu, Yanpei, et al. "Delving into transferable adversarial examples and black-box attacks." arXiv preprint arXiv:1611.02770 (2016).

Transferability-based attack

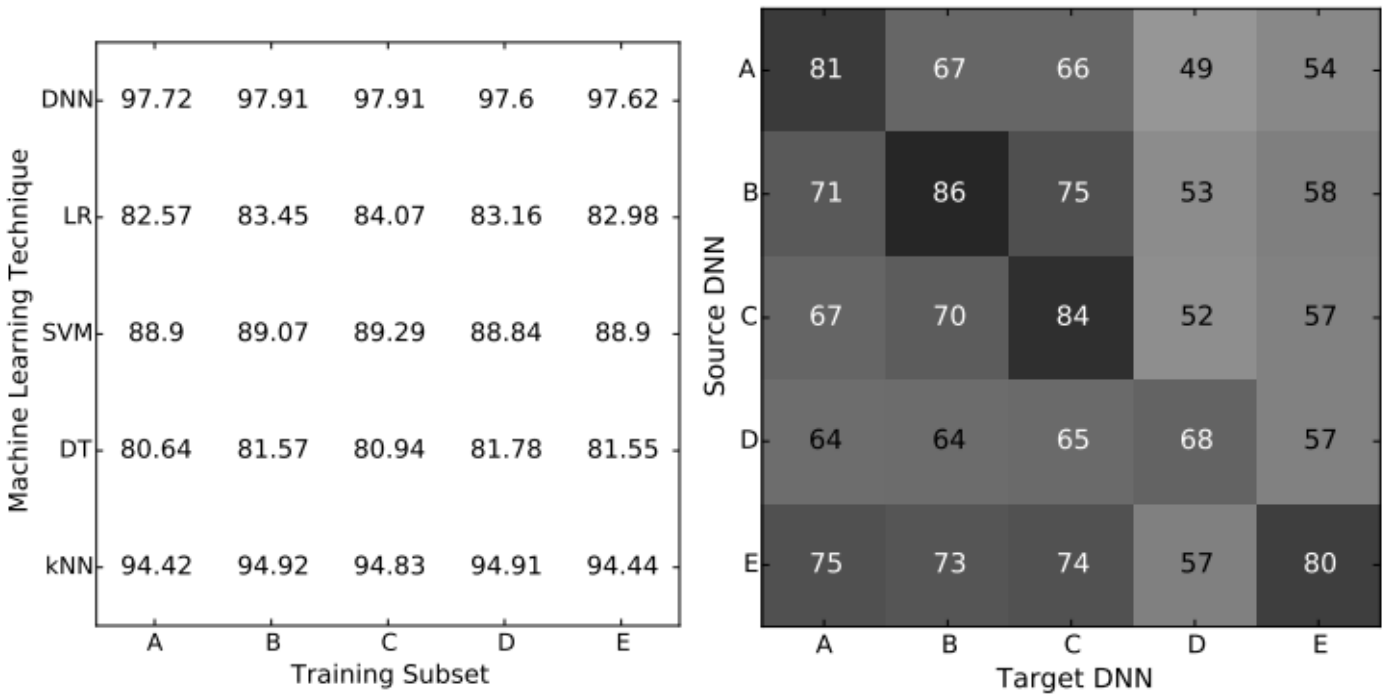
- Recall the definition of an adversarial example $\mathbf{x}'$ is $\mathbf{x}' = \mathbf{x} + \eta(\mathbf{x})$ where $\eta(\mathbf{x}) = \arg\min_{\mathbf{z}} f(\mathbf{x} + \mathbf{z}) \neq f(\mathbf{x})$.
- It is found in practice that model f's adversarial example $\mathbf{x}'$ often misleads another model f' as well.
- The notion of adversarial sample transferability is

$$\Omega_X(f, f') = |f'(\mathbf{x}) \neq f'(\mathbf{x} + \eta(\mathbf{x})), \mathbf{x} \in X|,$$

  where set X is s representative of the expected input distribution for the task solved by models f and f'.
- Thus, the larger $\Omega_X(f, f')$, the more likely an adversarial example $\mathbf{x}$ of f will also be an adversarial example of f'.
- If $\Omega_X(f, f')$ is large, black-box attacks on f' could be significantly improved by knowledge of f.
- Practically, transferability is classified into two case.
  - Intra-technique transferability.
  - Cross-technique Transferability.

Intra-technique Transferability

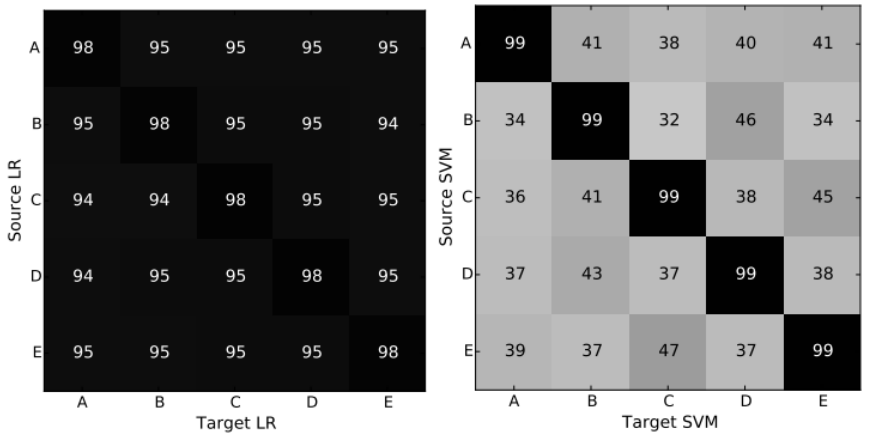
- Intra-technique transferability is defined across models, which are trained with the same machine learning technique but different parameter initializations or datasets.



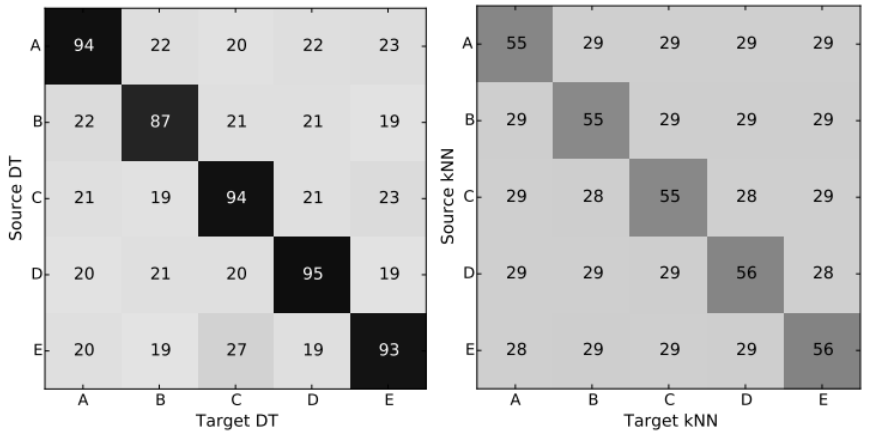
(a) Model Accuracies (b) DNN models

Figure 8: Intra-technique transferability for 5 ML techniques. (a) the accuracy rates of the 25 models used, computed on the MNIST test set.

Intra-technique Transferability



(c) LR models (d) SVM models



(e) DT models (f) kNN models

Figure 9: Intra-technique transferability for 5 ML techniques. (b-f) are such that cell (i, j) reports the intra-technique transferability between models i and j, i.e. the percentage of adversarial samples produced using model i misclassified by model j.

Intra-technique Transferability

- By observation, differentiable models like DNNs and LR are more vulnerable to intra-technique transferability than non-differentiable models like SVMs, DTs, and kNNs.
- In the case of SVMs, this could be explained by the explicit constraint during training on the choice of hyperplane decision boundaries that maximize the margins (i.e. support vectors).
- The robustness of both DTs and kNNs could simply stem from their non-differentiability.

Cross-technique Transferability

- Cross-technique transferability between models i and j, which are trained using different machine learning techniques, is defined as the proportion of adversarial samples produced to be misclassified by model i that are also misclassified by model j.

Cross-technique Transferability

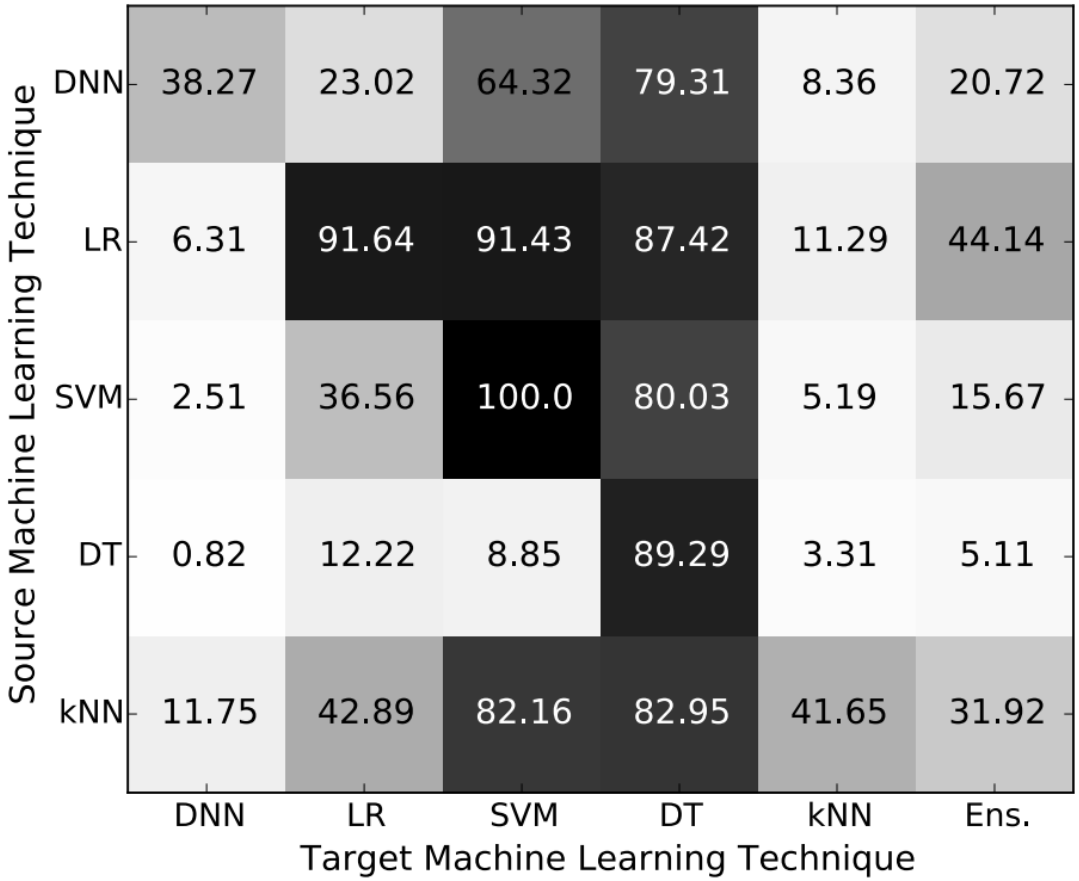


Figure 10: Cross-technique transferability matrix: cell (i, j) is the percentage of adversarial samples crafted to mislead a classifier learned using machine learning technique i that are misclassified by a classifier trained with technique j.

Cross-technique Transferability

- Cross-technique transferability is a strong phenomenon to which techniques like LR, SVM, DT, and ensembles are vulnerable, making it easy for attackers to craft adversarial samples misclassified by models trained using diverse machine learning techniques.

Black-box Attack

- Inspired by the notion of transferability, black-box attack can be achieved by the following three steps.
  - First, find/construct a substitutes f' of the unknown model f.
  - Second, generate adversarial examples on f'.
  - Third, use f''s adversarial examples to attack f.
- The quality of f is essential. Ideally, $f' = f$, which is unattainable.
- Picking of f' could be guided by the empirical findings of intra-technique transferability and cross-technique transferability.
  - For example, when an image classifier is under attack, it's natural to guess that the classifier is a convolutional network. Intra-technique transferability is significant for neural network. Thus, we can design the substitute f' to be a network.

Discussion on Black-box Method

- Black-box attack's main difficulty comes from limited knowledge of the targeted model compared to white-box method.
- There are many aspects of this limitation, besides the setting of no knowledge instead of the input-output discussed above.
  - Does the attacker know the task the model is solving (input space, output space, defender cost)?
  - Does the attacker know the machine learning algorithm being used?
  - Details of the algorithm? (Neural net architecture, etc.)
  - Learned parameters of the model?

Outline

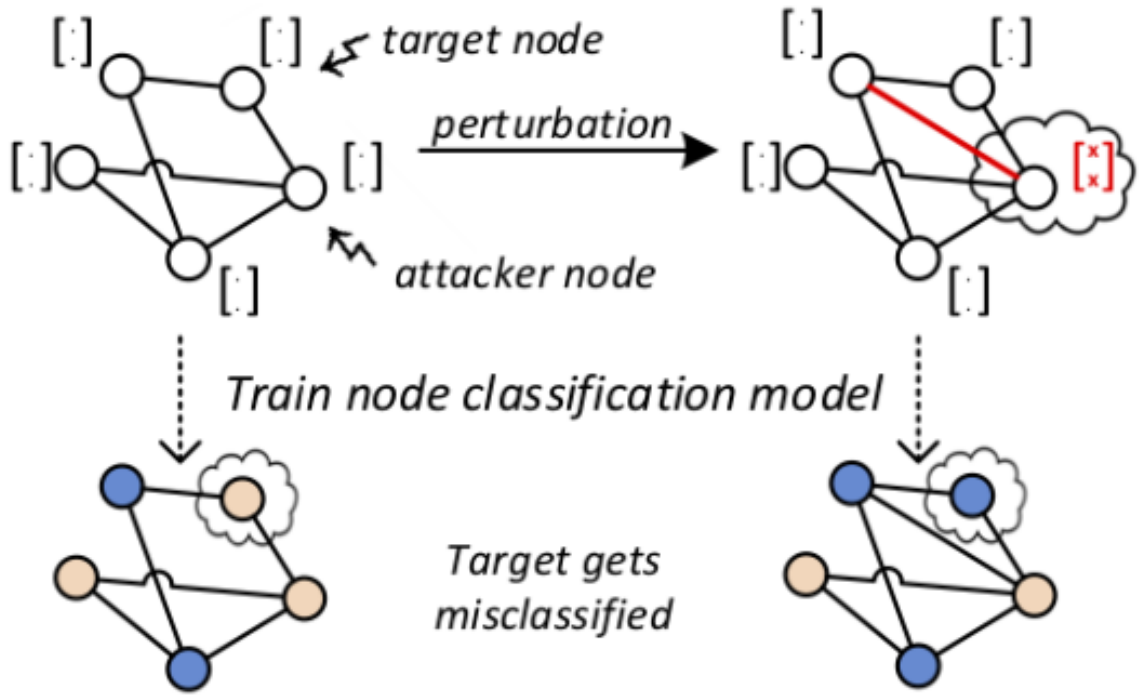
1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Graph Model Attack

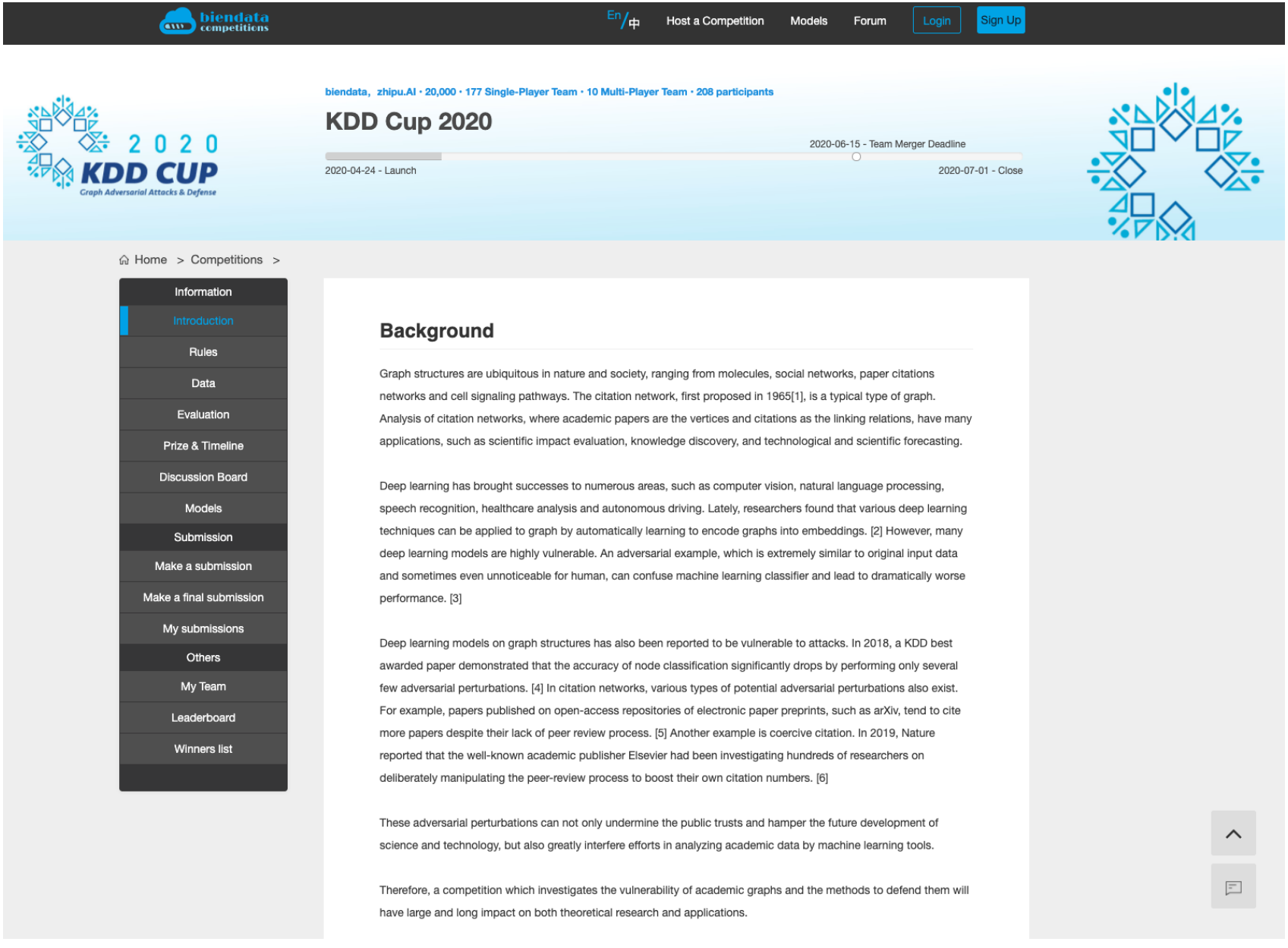
- Modify the graph structure and node features to cause GNN to give wrong predictions.
  - e.g. a bank needs to build a reliable credit evaluation system where their model should not be easily attacked by malicious manipulations
- Differences between attacking graph models and attacking traditional image classifiers:
  - Samples of the graph-structured data are not independent. Changing one's feature or connection will influence the prediction on others. Changing test data may even influence the training data so that the trained classifier is also changed.
  - When modifying the graph structure, the search space for adversarial example is discrete. Previous gradient methods may be invalid.

Greedy Method

- Formulated as a search for a perturbed graph G'.
- Use a heuristic algorithm to find the most influential operations on the graph.(e.g. removing/adding the edge or flipping the feature which can cause largest increase in the objective function)



Graph Attack Challenge



Greedy Method: Ensure Graph Similarity

- The perturbation should be unnoticeable, how to formally define the similarity between legitimate and adversarial graphs?
- Key constraints/maintainings [Zugner et al., 2018]:
  - the degree distribution should be maintained
  - two positive features which never happen together in G should also not happen together in G'
  - eigenvalues/eigenvectors of the graph Laplacian Matrix should also be maintained during attacking, otherwise the attacks are easily detected.[7]

[7] Ma et al., 2019. https://arxiv.org/abs/1906.03750.

Reinforcement Learning Method

- Only consider adding/removing edges.
- The attacker is allowed to modify m edges in total to alter F's prediction on the victim node v_0. The attacking mission is formulated as a Q-Learning game, with the defined Markov Decision Process:
  - **State**: The state s_t is represented by $(G^{(t)}, v_0)$, where $G^{(t)}$ is the modified graph within t iterative steps
  - **Action**: $a_t \in \mathcal{V} \times \mathcal{V}$ which means the edge to be added/removed
  - **Reward**: A positive reward if v_0's label is altered

$$r(s_m, a_m) = \begin{cases} 1 & \text{if } C(v_0, G^{(m)}) \neq y \\ -1 & \text{if } C(v_0, G^{(m)}) = y \end{cases}$$

  - **Termination**: Stop when the agent finishes modifying m edges

Graph Model Attack

- Graph Structure Poisoning via Meta-Learning
  - Use meta learning technique which takes the graph structure as the hyperparameter of the GCN model to optimize.
  - Perturbing 5% edges of a CITESEER graph dataset can increase the misclassification rate to over 30%
  - [Zugner & Gunnemann, 2019]
- Attack on Node Embedding
  - Perturb graph to corrupt the quality of node embedding
  - [Bojcheski &Gunnemann, 2018]
- RL with rewriting operation
  - Define a rewriting operation as $a = (v_1, v_2, v_3)$ where (v_1, v_2) removed and (v_1, v_3) added [Ma et al., 2019]
- MGA: Momentum Gradient Attack on Network
  - Integrate the momentum term into the iterative process to stabilize the updating direction, and achieve more aggressive attacks with fewer rewiring links [Chen et al., 2020]

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Attacks Beyond Classification

- Attacks on Autoencoders and Generative Models
  - Adversarial input image misleads the autoencoder to reconstruct a completely different image [Tabacof et al. 2016]
  - Adversarial attacks for deep generative models, e.g. VAE&VAE-GAN [Kos et al. 2017]
  - e.g. In MNIST dataset, the input image is "1" while the reconstructed image is "0".
- Attacks on Deep Reinforcement Learning
  - Adversarial attacks luring the agents [Lin et al. 2017]
  - FGSM for degrading performance of trained policies [Huang et al. 2017]
    - take one-step gradient on the state x to craft a fake state x'.
    - a slight perturbation on RL model's state can cause large difference on decision and performance.

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Defense

- There's no absolutely secure system in this world.
- Attacks are always possible.
- The main concern of both attackers and defenders are the cost, either of attacking or of defense.

Defense

- The most simple defense strategy is doing data augmentation.
- For a trained model $f(X)$ and its adversarial examples' set X', train a model f on $X + X'$ would result in a more resilient model.
- Why? Because we force the model to see adversarial examples so it can handle them in the future.
- A drawback is X' can be very large, yet X' can never be completed. The resulted model will have its own adversarial examples again.

Defense

- Another idea is to inject adversarial examples as the training goes, continually generating new adversarial examples at every step of training.
- This leads to the following loss

$$L = \frac{1}{m - k + \lambda k}\left(\sum_{i \in CLEAN} L(X_i|y_i) + \lambda \sum_{i \in ADV} L(X_i^{adv}|y_i)\right). \quad (2)$$

  In other words, in every step of training, the adversarial examples generated from last step is forced to be correctly classified in this step.
- This is more adjustable compared to simple data-augmentation.

Discussion on Defense

- These defense strategies assume that attacking strategies are known, thus the adversarial examples are available at training stage. What if attackers change their strategy? Besides, real life is full of surprise.
- One question related is whether there is a universal defense strategy, at least to some extent.
- When attackers are adaptive/reactive, defense becomes extremely difficult. It's a long way to solve this problem.

Adversarial Learning Competition

- Adversarial learning has become a hot topic and will become even hotter as more ML applications are applied in real life.
- NIPS2017 held an Adversarial Learning Competition.
- The overall goal of the challenge is to facilitate more generally applicable adversarial attacks and defenses.
- Adversarial Learning Competition has two tracks.
  - Attack track: add noise to input images with malicious intent to convert them to adversarial examples.
  - Defense track: defend against adversarial examples generated by the attack track.
- Check here to check more details.

Outline

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Conclusion

- Machine learning models are vulnerable to attacks.
- Attackers can leverage both the information of the model(white-box attack) and the prior(black-box attack) to attack a model.
- Defense is possible but hard in real life.

Overview

1. Introduction
2. White-box Attack
3. Black-box Attack
4. Graph Model Attack
5. Attacks Beyond Classification
6. Defense
7. Conclusion

Thanks.

HP: http://keg.cs.tsinghua.edu.cn/jietang/
Email: jietang@tsinghua.edu.cn